# Module Interface Specification for RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

November 10, 2025

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| Date 1 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

# 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at:
https://github.com/ZifanSi/vision-guided-tracker/blob/main/docs/SRS/SRS.pdf

# Contents

# 3 Introduction

The following document details the Module Interface Specifications for Rocam: High Performance Vision-Guided Rocket Tracker.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at:

https://github.com/ZifanSi/vision-guided-tracker

# 4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by RoCam.

| **Data Type** | **Notation** | **Description** |
|---|---|---|
| character | char | a single symbol or digit |
| integer | $\mathbb{Z}$ | a number without a fractional component in (-$\infty$, $\infty$) |
| natural number | $\mathbb{N}$ | a number without a fractional component in [1, $\infty$) |
| real | $\mathbb{R}$ | any number in (-$\infty$, $\infty$) |

The specification of RoCam uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, RoCam uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

# 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

| **Level 1** | **Level 2** |
|---|---|
| Jetson Module | Gimbal Abstraction Module<br>Computer Vision Module<br>Tracking Module<br>Output Video Module<br>Recording Module<br>State Management Module<br>API Gateway Module |
| UI Module | Preview Module<br>Manual Control Module<br>Recording Management Module<br>Configuration Module |

Table 1: Module Hierarchy

# 6 MIS of Gimbal Abstraction Module

## 6.1 Module

Gimbal

## 6.2 Uses

This module does not use any other modules.

## 6.3 Syntax

### 6.3.1 Exported Constants

This module does not have any exported constants.

### 6.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| move_deg | tilt: f32, pan: f32 | - | gimbalCommunicationError |
| measure_deg | - | tilt: f32, pan: f32 | gimbalCommunicationError |
| control_arm_led | enabled: bool | - | gimbalCommunicationError |
| control_status_led | enabled: bool | - | gimbalCommunicationError |

## 6.4 Semantics

### 6.4.1 State Variables

- presistent connection to the gimbal

### 6.4.2 Environment Variables

- This module interacts with an external gimbal device.

### 6.4.3 Assumptions

### 6.4.4 Access Routine Semantics

move_deg(tilt: f32, pan: f32):

- transition: sends the tilt and pan angles to the gimbal
- output: None
- exception: gimbalCommunicationError

measure_deg():

- transition: retrieves the tilt and pan angle measurements from the gimbal

- output: (tilt: f32, pan: f32)
- exception: gimbalCommunicationError

control_arm_led(enabled: bool):

- transition: controls the arm LED on the gimbal
- output: None
- exception: gimbalCommunicationError

control_status_led(enabled: bool):

- transition: controls the status LED on the gimbal
- output: None
- exception: gimbalCommunicationError

### 6.4.5 Local Functions

# 7 MIS of Computer Vision Module

## 7.1 Module

CV

## 7.2 Uses

This module does not use any other modules.

## 7.3 Syntax

### 7.3.1 Exported Constants

- WIDTH: width of the video feed (in pixels)
- HEIGHT: height of the video feed (in pixels)

### 7.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| get_frame | - | frame | cameraError |
| get_rocket_location | frame | x: f32, y: f32 | noRocketFound, cameraError |

### 7.4 Semantics

#### 7.4.1 State Variables

- computer vision model
- presistent connection to the camera sensor

#### 7.4.2 Environment Variables

- This module interacts with an external camera sensor.

#### 7.4.3 Assumptions

#### 7.4.4 Access Routine Semantics

get_frame():

- transition: retrieves a frame from the camera sensor
- output: frame
- exception: cameraError

get_rocket_location(frame):

- transition: analyzes the frame to find the rocket
- output: (x: f32, y: f32)
- exception: noRocketFound, cameraError

#### 7.4.5 Local Functions

## 8 MIS of Tracking Module

### 8.1 Module

Tracking

### 8.2 Uses

This module uses the Computer Vision Module (7) and the Gimbal Abstraction Module (6).

### 8.3 Syntax

#### 8.3.1 Exported Constants

This module does not have any exported constants.

#### 8.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| step | - | - | - |

### 8.4 Semantics

#### 8.4.1 State Variables

- PID parameters for controlling the gimbal

#### 8.4.2 Environment Variables

None

#### 8.4.3 Assumptions

#### 8.4.4 Access Routine Semantics

step():

- transition:
    1. Retrieve the current gimbal angle from the Gimbal Abstraction Module.
    2. Get the location of the rocket from the Computer Vision Module.
    3. Calculate the desired gimbal angle using the PID algorithm.
    4. Send the desired gimbal angle to the Gimbal Abstraction Module.
- output: None
- exception: None

#### 8.4.5 Local Functions

## 9 MIS of Output Video Module

### 9.1 Module

videoOut

### 9.2 Uses

### 9.3 Syntax

#### 9.3.1 Exported Constants

#### 9.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| output_frame | states | - | outputDeviceError |

### 9.4 Semantics

#### 9.4.1 State Variables

None

#### 9.4.2 Environment Variables

This module displays the frame on the screen connected to the Jetson.

#### 9.4.3 Assumptions

#### 9.4.4 Access Routine Semantics

output_frame(states):

- transition:
    1. Retrieve the frame from the Computer Vision Module.
    2. Retrieve the location of the rocket from the Computer Vision Module.
    3. Crop the frame so the rocket is in the center of the frame.
    4. Overlay the states on the cropped frame.
    5. Display the cropped frame on the screen connected to the Jetson.
- output: None
- exception: outputDeviceError

#### 9.4.5 Local Functions

## 10 MIS of Recording Module

### 10.1 Module

recording

### 10.2 Uses

### 10.3 Syntax

#### 10.3.1 Exported Constants

#### 10.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| start_recording | - | - | recordingError |
| stop_recording | - | - | recordingError |
| list_recordings | - | recordings: list[Recording] | |
| delete_recording | recordingId: str | - | |

## 10.4 Semantics

### 10.4.1 State Variables

- recording status

### 10.4.2 Environment Variables

This module interacts with the file system to record the video and log files.

### 10.4.3 Assumptions

### 10.4.4 Access Routine Semantics

start_recording():

- transition: starts recording the video and log files
- output: None
- exception: recordingError

stop_recording():

- transition: stops recording the video and log files
- output: None
- exception: recordingError

list_recordings():

- transition: retrieves the list of recordings from the file system
- output: recordings: list[Recording]
- exception: None

delete_recording(recordingId: str):

- transition: deletes the recording from the file system
- output: None
- exception: None

### 10.4.5 Local Functions

# 11 MIS of State Management Module

## 11.1 Module

stateManagement

## 11.2 Uses

This module uses the Recording Module (10), Tracking Module (8), and Output Video Module (9).

## 11.3 Syntax

### 11.3.1 Exported Constants

This module does not have any exported constants.

### 11.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| arm | - | - | - |
| disarm | - | - | - |
| manual_control | direction | - | - |

## 11.4 Semantics

### 11.4.1 State Variables

- armed status

### 11.4.2 Environment Variables

None

### 11.4.3 Assumptions

### 11.4.4 Access Routine Semantics

arm():

- transition:
  1. Set the armed state variable to true.
  2. Start a loop in the background, which does the following until the armed state variable is set to false:
     (a) Call "get_frame" from the Computer Vision Module.
     (b) Call "get_rocket_location" from the Computer Vision Module.
     (c) Call "step" from the Tracking Module to adjust the gimbal to keep the rocket in the center of the frame.
     (d) Call "output_frame" from the Output Video Module to display the frame on the screen.
- output: None

- exception: None

disarm():

- transition: sets the armed state variable to false
- output: None
- exception: None

manual_control(direction):

- transition: adjusts the gimbal to the given direction if the armed state variable is false
- output: None
- exception: None

#### 11.4.5 Local Functions

# 12 MIS of API Gateway Module

## 12.1 Module

apiGateway

## 12.2 Uses

This module uses the State Management Module (11) .

## 12.3 Syntax

### 12.3.1 Exported Constants

This module does not have any exported constants.

### 12.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| start_server | - | - | - |

## 12.4 Semantics

### 12.4.1 State Variables

None

### 12.4.2 Environment Variables

None

#### 12.4.3 Assumptions

#### 12.4.4 Access Routine Semantics

start_server():

- transition: starts the api server and listens for requests from the web interface.
- output: None
- exception: None

#### 12.4.5 Local Functions

# 13 MIS of Preview Module

## 13.1 Module

preview

## 13.2 Uses

This module uses the API Gateway Module (12).

## 13.3 Syntax

### 13.3.1 Exported Constants

This module does not have any exported constants.

### 13.3.2 Exported Access Programs

This module does not have any exported access programs

## 13.4 Semantics

### 13.4.1 State Variables

None

### 13.4.2 Environment Variables

- This module shows the preview on the screen.

#### 13.4.3 Assumptions

#### 13.4.4 Access Routine Semantics

#### 13.4.5 Local Functions

# 14 MIS of Manual Control Module

## 14.1 Module

manualControl

## 14.2 Uses

This module uses the State Management Module (11).

## 14.3 Syntax

### 14.3.1 Exported Constants

This module does not have any exported constants.

### 14.3.2 Exported Access Programs

This module does not have any exported access programs

## 14.4 Semantics

### 14.4.1 State Variables

None

### 14.4.2 Environment Variables

- This module shows the manual control interface on the screen.

### 14.4.3 Assumptions

### 14.4.4 Access Routine Semantics

### 14.4.5 Local Functions

# 15 MIS of Recording Management Module

## 15.1 Module

recordingManagement

## 15.2 Uses

This module uses the Recording Module (10).

## 15.3 Syntax

### 15.3.1 Exported Constants

This module does not have any exported constants.

### 15.3.2 Exported Access Programs

This module does not have any exported access programs

## 15.4 Semantics

### 15.4.1 State Variables

- recording list

### 15.4.2 Environment Variables

- This module manages the recordings.

### 15.4.3 Assumptions

### 15.4.4 Access Routine Semantics

### 15.4.5 Local Functions

# 16 MIS of Configuration Module

## 16.1 Module

configuration

## 16.2 Uses

This module uses the Recording Management Module (15).

## 16.3 Syntax

### 16.3.1 Exported Constants

This module does not have any exported constants.

### 16.3.2 Exported Access Programs

This module does not have any exported access programs

## 16.4 Semantics

### 16.4.1 State Variables

- configuration settings

### 16.4.2 Environment Variables

- This module manages the configuration settings.

### 16.4.3 Assumptions

### 16.4.4 Access Routine Semantics

### 16.4.5 Local Functions

# References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

# 17 Appendix

# Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?
4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?
5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)
6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)